

With a modest number of components, the standard PC printer port can be transformed into an I/O system with up to 256 I/O lines. This article completely describes both the hardware and the necessary BASIC software. The hardware can also easily be controlled using other programming languages.

Design by V. Himpe

printer port extension

up to 256 I/O lines on a standard printer port

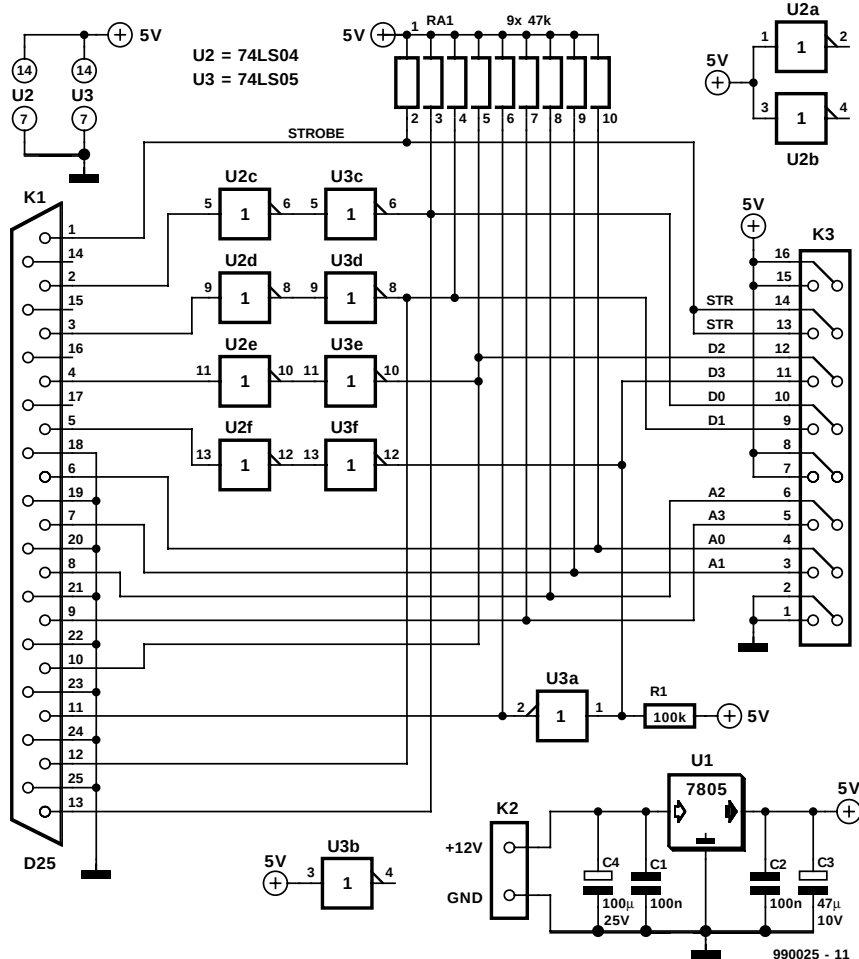


Figure 1. The schematic diagram of the bus unit, which is the heart of the I/O system.

It can hardly be said often enough that the PC was primarily developed to meet the needs of the office environment. This basically means that in the overall design, a lot of attention has been given to connecting standard peripheral equipment, such as printers and modems. Analogue and/or digital inputs and outputs, which would be useful to the user who wants to perform control tasks, are thus simply not present in a standard PC configuration.

The circuit that is presented here, or better said the complete system, was originally named 'Paraport' by its developer, since it is an I/O extension for the parallel port. It provides a large number of digital I/O lines for a normal PC at a modest cost. If you decide on the maximum configuration, this amounts to 256 lines. Although it might seem natural to arrange these as 128 inputs and 128 outputs, you are free to divide the lines between inputs and outputs as you desire.

The inputs work with TTL levels, while the outputs have buffers with open-collector outputs, so that they can directly switch relatively heavy loads. Thanks to the modular nature of the design, you can build exactly the amount of hardware that is needed to satisfy your specific needs. There is thus no need for any unnecessary investment, and the hardware can be kept as compact as possible.

Design approach

As already mentioned, this project is based on a modular design. There are

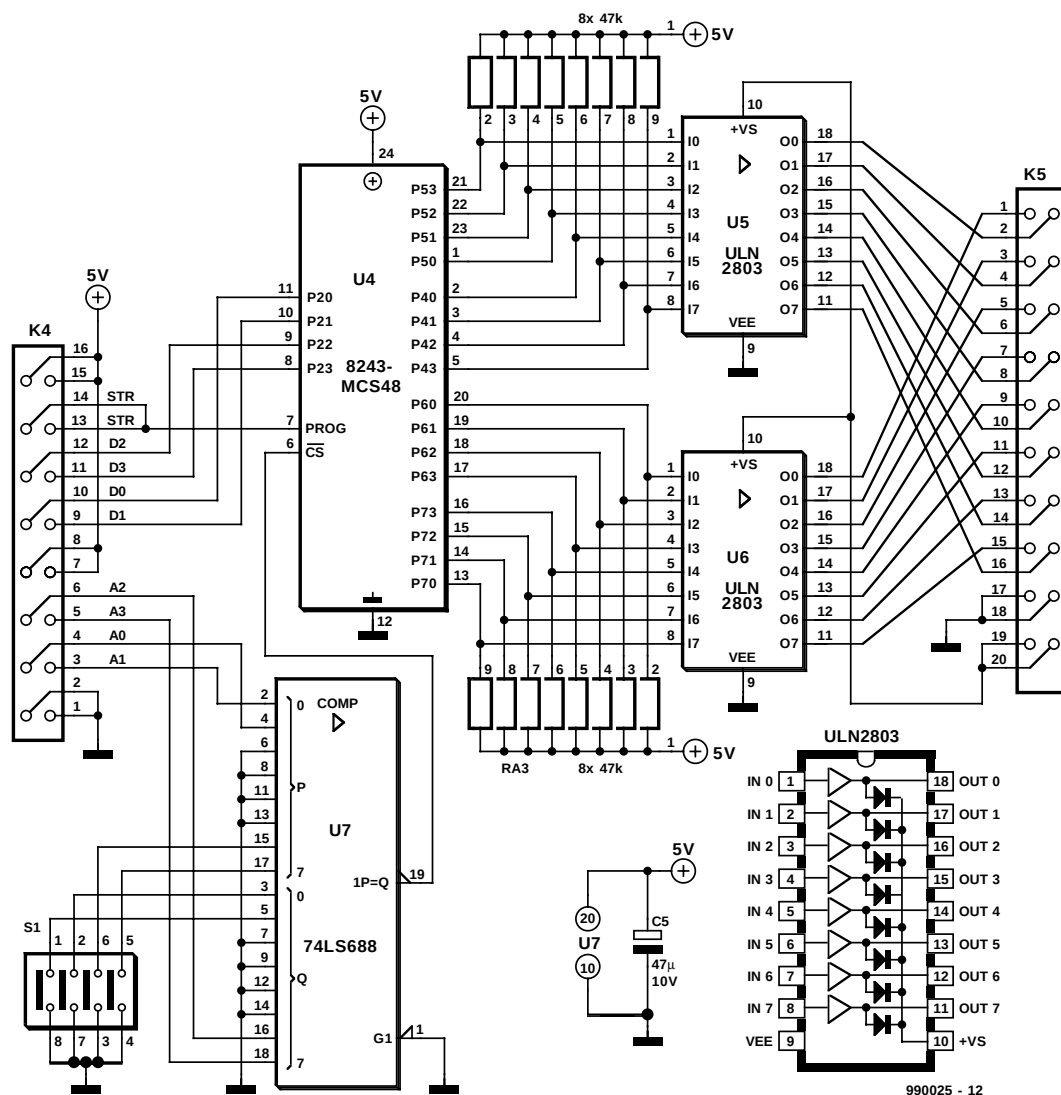


Figure 2. Up to 16 of the I/O units shown here can be connected to the bus unit.

two types of units: an interface to the printer port (the bus unit) and the actual I/O unit. Each I/O unit provides eight TTL-compatible inputs and 8 buffered outputs. Up to 16 I/O units can be connected to the bus unit. In the maximum configuration, a total of 17 circuit boards are thus connected to the printer port.

The hardware

Figure 1 shows the design of the bus unit. This circuit contains a number of inverters that adequately buffer the printer port signals. Since the standard printer port is intended to function primarily as an output port, recourse is made to several control lines of the port for input data. The data lines D0 through D3 from the PC are used for data output, and the control lines Acknowledge, Busy, Paper Empty and Select are used for data input. The remaining four outputs (D4 through D7) are used to address the sixteen I/O modules. Finally, the Strobe signal has

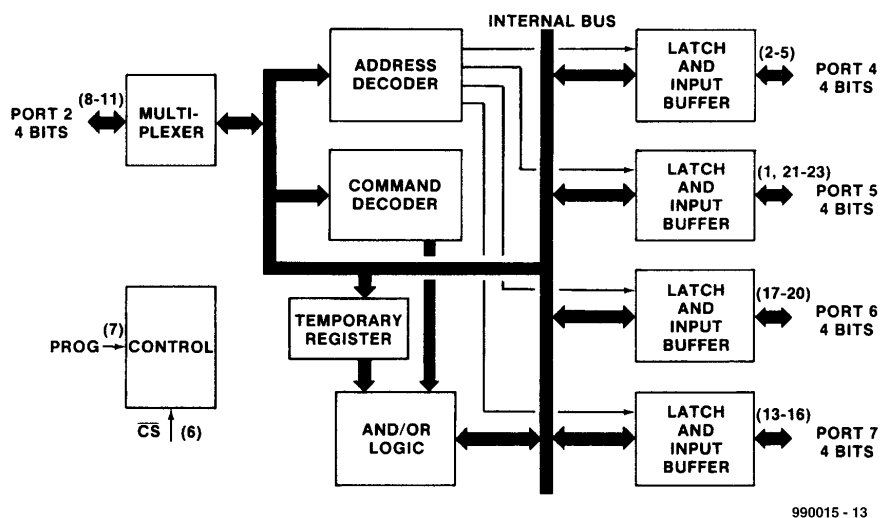


Figure 3. The internal structure of the 82C43, a controller with five 4-bit ports.

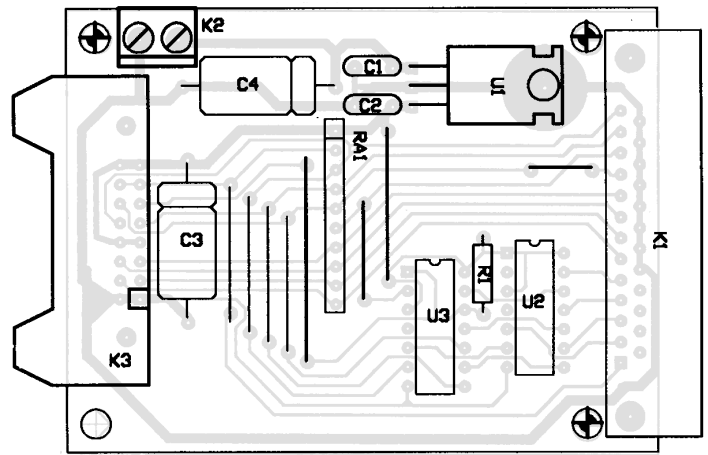
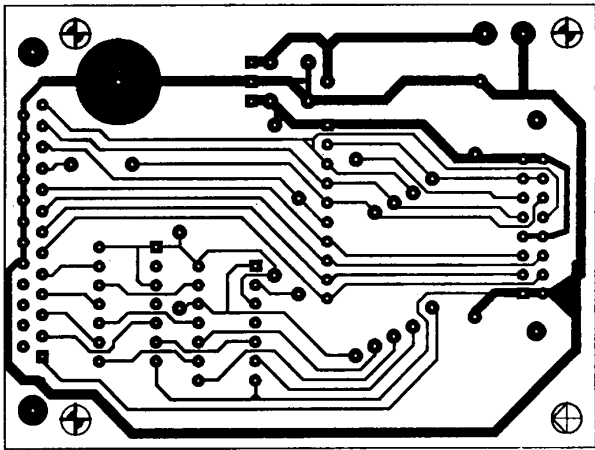


Figure 4. The copper-track layout and component overlay for the bus unit card, as made by the author.

COMPONENTS LIST

bus-unit

Resistors:

R1 = 47k Ω
RA1 = SIL array 9 x 47 k Ω

Capacitors:

C1,C2 = 100nF
C3 = 100 μ F 25V
C4 = 47 μ F 25V

Semiconductors:

U1 = 7805
U2 = 74LS04
U3 = 74LS05

Miscellaneous:

K1 = 25-pin angled sub-D connector, male, PCB mount
K2 = 2-way PCB terminal block
K3 = 16-pin boxheader, angled, with eject levers

also been assigned a function. It ensures that the I/O system reads the data on the D0 through D3 lines at the proper time; these lines are used for both data and commands. Only one such bus unit per printer port is needed for this project.

Modular design

The I/O units are connected to the bus unit. **Figure 2** shows the design of a single I/O unit. The heart of this unit is an 82C43 IC, which is a port expander chip that was specifically developed by Intel to expand the I/O capabilities of its 8048 processor (the predecessor of the 8041/51 processors). For those of you that are not familiar with the operation of the 82C43, a short explanation is in order. The IC is supplied in a 24-pin package and contains five 4-bit bi-directional ports (labelled P2, P4, P5, P6 and P7) and two control inputs (PROG and CS). **Figure 3** shows the internal organization of the 82C43. The

COMPONENTS LIST

I/O-unit (up to 16 off)

Resistors:

RA1,RA2 = SIL array 8 x 47k Ω
U5,U6 = DIL array 8x330 Ω *

Capacitors:

C5 = 10 μ F 25V

Semiconductors:

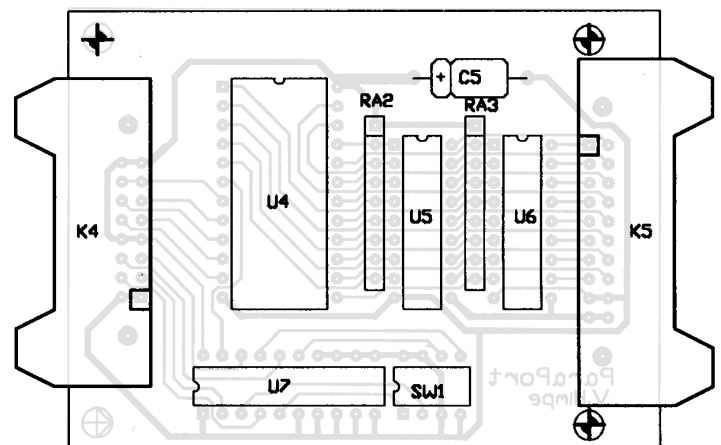
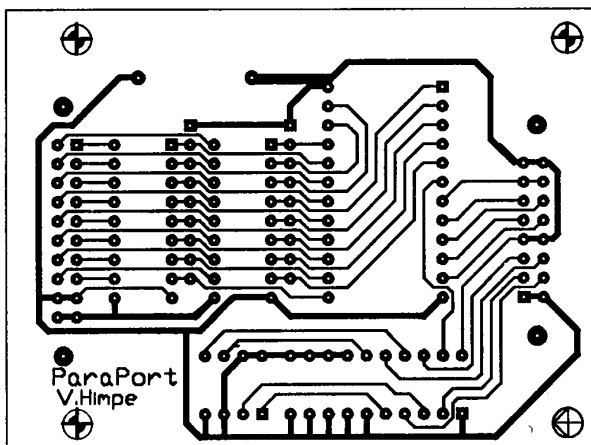
U4 = 82C43 or 8243 (Intel)
U5,U6 = ULN2803*
U7 = 74LS688

Miscellaneous:

K4 = 16-pin boxheader, angled, with eject levers
K5 = 20-pin boxheader, angled, with eject levers
SW1 = 4-way DIL switch

* see text

Figure 5. The copper-track layout and component overlay for the I/O unit.



port numbering may seem to be a bit strange, but this is due to the fact that this IC was intended to be used in combination with an 8048 microcontroller, which already has several ports of its own.

One of the 4-bit ports (P2) is used for communication with the printer port, and is thus connected directly to the printer output lines D0 through D3. This leaves 16 lines free for I/O functions. The Strobe signal from the printer port is connected directly to the PROG input of the 82C43, and thus controls data input. The remaining control input of the 82C43 (CS) is connected to an address decoder based on a 74LS688.

The 74LS688 is an old acquaintance. It is a word comparator that is very often used in address decoders, as in this project. The module address is selected by the 4-pole DIP switch SW1, in the range of 0 through 15. If the address selected by SW1 is equal to the value present on the address bus, the decoder output goes low, which selects the I/O unit in question.

As can be seen from the schematic diagram, every I/O line has the same structure. This means that every line can be used as an input line or an output line, as desired. However, the selection (either input or output) must be made for a set of eight lines at a time, since the buffers used here are packaged eight to a chip.

If a set of lines is to be used as inputs, the associated integrated buffer (ULN2803) must be removed and replaced by an 8-position, 330-Ω resistor array. This array must be housed in a DIL package, so that it can be plugged into the socket provided for the buffer IC. When inserting the resistor array, you must be sure that all inputs are connected to their associated outputs. They are all arranged directly across from each other, so the only thing you need to do is to take care that the resistor array is properly inserted in the socket. When it is correctly placed, pin positions 9 and 10 of the socket will be open. The pull-up resistor array ensures that open inputs are held at a defined logic level. Since the inputs are only suitable for TTL signals, the actual input signals must naturally also conform to this standard.

If a set of lines is to be used as outputs, the associated ULN2803 must be installed. This IC has open-collector outputs. Since the loads connected to the outputs may be inductive, each output has a built-in protection diode. The cathodes of all these diodes are

```
' REM PARAPORT programming example
' Written by Vincent Himpe

' printerport initialisation
' LPT1= &h378 &h379 &h37A
' LPT2= &h278 &h279 &h27A
' PRN = &h3BC &h3BD &h3BE

' controller instruction set
' read      = 0
' write     = 4
' Write_or  = 8
' Write_and = 12

' ports numbered 0 through 3 :
' this corresponds to chip ports
' 4 through 7 (pin names)

***** initialisation *****
OUT &h378,255 ' set all to 1
OUT &h37A,0   ' pull strobe high

***** Write *****

' formula to compute command word:
' (Cardaddress *16)+command+port
' example : write to port 2 on card 1
' (1*16)+ 4 +1 = 21

out &h378,21 ' card address + port address + command
out &h37A,1  ' strobe low

' formula to compute data:
' (Cardaddress*16)+data
' so in this case (1*16)+data
' for testing we make the 2 last bits high  HLL = 12
' so the result becomes : (1*16)+12 = 28

out &h378,28 ' prepare data
out &h37A,0  ' strobe high

' data now available on desired port

' ***** Read *****
' Command formula: same as with writing, but using diff. command
' (1*16)+0+1 = 17

out &h378,17 ' card address + port address + command
out &h37A,1  ' strobe low
out &h378,255 ' set controller outputs to high-Z

value = inp(&h379) ' read
value = ((value and &hF0)/16) ' extract highest nibble
out &h37A,0       ' strobe high

' ***** end *****
```

Figure 6. A BASIC program that can be used to control the I/O system.

connected to the V_p pin. They can thus be connected to the highest-available potential via connector K5. The earth pin is connected to the system earth. Each buffer can switch voltages up to 50 V and currents up to 500 mA, which is more than adequate for most applications.

Compact hardware

Figure 4 shows the copper-track layout and the component overlay of the

printed circuit board for the bus unit, while Figure 5 shows the same for the I/O unit. Both types of circuit boards are single-sided, so that you can easily make them yourself. The price of this approach is that several wire jumpers are needed on the bus unit card. Fortunately, only one of these has to be built, so this is not a particularly onerous disadvantage.

A length of flat cable can be used to connect the I/O units to the bus unit, since headers are used for the con-

nectors on the bus side (K3 and K4). If the I/O cards are mounted one on top of the other using 10-mm spacers, the whole system can be constructed as a compact and sturdy assembly. The desired address for each I/O card must be selected using the associated DIP switch; each unit must have its own (unique) address.

A mains adapter with an 12-V DC output can be used to power the system. The 5-V supply voltage produced in the bus unit is provided to the various I/O units via the bus cable.

A few lines of BASIC

Most hardware also needs some kind of software, and that is true for this project. Before delving into the software, it's necessary to give a little more attention to the 82C43. Since this IC was specifically developed to work with an 8048 controller, a special procedure must be used to write to and read from its registers. This approach, which allows data to be read into or written from the 8048 with a single

Table 1. Port selection.

input/output address code		selected input/output
P21 (D1)	P20 (D0)	
0	0	4
0	1	5
1	0	6
1	1	7

instruction, can also be used to advantage here.

The 4-bit bus can be used for both data and commands. To send a command to the IC, the command bits must be placed on the bus and the level of the Strobe line then switched from high to low. Data can then be transferred by placing the data bits on the bus and switching the Strobe line back to high.

The 82C43 recognizes four basic commands: Read, Write, Write_AND and Write_OR. Logically enough, Read is used to read data and Write to write data, so these commands need no further explanation. With Write_AND, the new data written to the IC are log-

Table 2. Command selection.

Command		Function
P23 (D3)	P22 (D2)	
0	0	Read
0	1	Write
1	0	ORLD
1	1	ANLD

ically ANDed with the data that are already present in the selected port, and the result is retained in the port. With Write_OR a similar process takes place, except that a logical OR is performed instead of a logical AND. The manner in which the various ports are selected and the way that the various functions are activated are described in **Table 1** and **Table 2**, respectively. As can be clearly seen from the program listing in Figure 6, it is not at all difficult to work with this circuit in actual practice. If you study this simple program, you will learn all that you need to know to be able to use all functions of the I/O system.